

© 2006 Blackwell Publishing Ltd, *Journal of Internal Medicine* 260: 103–110

- PC, only 12
- cheap, fast and robust software
- PCs
- hard to subvert: distributed, no central authority
- Performance: high-throughput, no limits
- all participants are equally important
- Chen et al. (2005, 2006, 2007)

- `testComplex` - `Test` object
- `TestComplex` - `Test` object
- `get_val` - `int` type value without parentheses
- `set_val` - `Test` object
- `get_val` - `Test` object
- `TestComplex` - `Test` object

```

1  # 计算欧氏距离
2  def euclidean_distance(x1, x2):
3      return ((x1[0] - x2[0])**2 + (x1[1] - x2[1])**2)**0.5
4
5  # 计算曼哈顿距离
6  def manhattan_distance(x1, x2):
7      return abs(x1[0] - x2[0]) + abs(x1[1] - x2[1])
8
9  # 计算切比雪夫距离
10 def chebyshev_distance(x1, x2):
11     return max(abs(x1[0] - x2[0]), abs(x1[1] - x2[1]))
12
13 # 计算闵可夫斯基距离
14 def minkowski_distance(x1, x2, p):
15     return ((abs(x1[0] - x2[0])**p + abs(x1[1] - x2[1])**p)**(1/p))
16
17 # 计算马氏距离
18 def mahalanobis_distance(x1, x2, cov):
19     diff = x1 - x2
20     return diff.T * cov * diff
21
22 # 计算余弦距离
23 def cosine_distance(x1, x2):
24     dot = x1[0]*x2[0] + x1[1]*x2[1]
25     norm1 = (x1[0]**2 + x1[1]**2)**0.5
26     norm2 = (x2[0]**2 + x2[1]**2)**0.5
27     return 1 - dot / (norm1 * norm2)
28
29 # 计算汉明距离
30 def hamming_distance(x1, x2):
31     return sum(x1[i] != x2[i] for i in range(len(x1)))
32
33 # 计算编辑距离
34 def edit_distance(x1, x2):
35     m, n = len(x1), len(x2)
36     dp = [[0] * (n + 1) for _ in range(m + 1)]
37     for i in range(1, m + 1):
38         for j in range(1, n + 1):
39             if x1[i - 1] == x2[j - 1]:
40                 dp[i][j] = dp[i - 1][j - 1]
41             else:
42                 dp[i][j] = min(dp[i - 1][j], dp[i][j - 1], dp[i - 1][j - 1]) + 1
43     return dp[m][n]
44
45 # 计算杰卡德距离
46 def jaccard_distance(x1, x2):
47     set1 = set(x1)
48     set2 = set(x2)
49     intersection = len(set1 & set2)
50     union = len(set1 | set2)
51     return 1 - intersection / union
52
53 # 计算皮尔逊距离
54 def pearson_distance(x1, x2):
55     mean1 = sum(x1) / len(x1)
56     mean2 = sum(x2) / len(x2)
57     cov = sum((x1[i] - mean1) * (x2[i] - mean2) for i in range(len(x1))) / len(x1)
58     var1 = sum((x1[i] - mean1)**2 for i in range(len(x1))) / len(x1)
59     var2 = sum((x2[i] - mean2)**2 for i in range(len(x2))) / len(x2)
60     return 1 - cov / (var1 * var2)**0.5
61
62 # 计算相关系数
63 def correlation_coefficient(x1, x2):
64     mean1 = sum(x1) / len(x1)
65     mean2 = sum(x2) / len(x2)
66     cov = sum((x1[i] - mean1) * (x2[i] - mean2) for i in range(len(x1))) / len(x1)
67     var1 = sum((x1[i] - mean1)**2 for i in range(len(x1))) / len(x1)
68     var2 = sum((x2[i] - mean2)**2 for i in range(len(x2))) / len(x2)
69     return cov / (var1 * var2)**0.5
70
71 # 计算余弦相似度
72 def cosine_similarity(x1, x2):
73     dot = x1[0]*x2[0] + x1[1]*x2[1]
74     norm1 = (x1[0]**2 + x1[1]**2)**0.5
75     norm2 = (x2[0]**2 + x2[1]**2)**0.5
76     return dot / (norm1 * norm2)
77
78 # 计算汉明相似度
79 def hamming_similarity(x1, x2):
80     return sum(x1[i] == x2[i] for i in range(len(x1))) / len(x1)
81
82 # 计算编辑相似度
83 def edit_similarity(x1, x2):
84     m, n = len(x1), len(x2)
85     dp = [[0] * (n + 1) for _ in range(m + 1)]
86     for i in range(1, m + 1):
87         for j in range(1, n + 1):
88             if x1[i - 1] == x2[j - 1]:
89                 dp[i][j] = dp[i - 1][j - 1] + 1
90             else:
91                 dp[i][j] = max(dp[i - 1][j], dp[i][j - 1], dp[i - 1][j - 1])
92     return dp[m][n] / max(m, n)
93
94 # 计算杰卡德相似度
95 def jaccard_similarity(x1, x2):
96     set1 = set(x1)
97     set2 = set(x2)
98     intersection = len(set1 & set2)
99     union = len(set1 | set2)
100    return intersection / union
101
102 # 计算皮尔逊相似度
103 def pearson_similarity(x1, x2):
104     mean1 = sum(x1) / len(x1)
105     mean2 = sum(x2) / len(x2)
106     cov = sum((x1[i] - mean1) * (x2[i] - mean2) for i in range(len(x1))) / len(x1)
107     var1 = sum((x1[i] - mean1)**2 for i in range(len(x1))) / len(x1)
108     var2 = sum((x2[i] - mean2)**2 for i in range(len(x2))) / len(x2)
109     return cov / (var1 * var2)**0.5
110
111 # 计算相关系数
112 def correlation_coefficient(x1, x2):
113     mean1 = sum(x1) / len(x1)
114     mean2 = sum(x2) / len(x2)
115     cov = sum((x1[i] - mean1) * (x2[i] - mean2) for i in range(len(x1))) / len(x1)
116     var1 = sum((x1[i] - mean1)**2 for i in range(len(x1))) / len(x1)
117     var2 = sum((x2[i] - mean2)**2 for i in range(len(x2))) / len(x2)
118     return cov / (var1 * var2)**0.5

```

The diagram illustrates the system architecture. At the top, three 'CPU' blocks are shown, each with a label 'CPU' and a small icon. These are connected to a central 'CPU' block. Below this central 'CPU' block is a 'Memory' block. A horizontal line separates this top section from the bottom section. In the bottom section, a 'USB' block is connected to a 'CPU' block, which is then connected to three 'GPU' blocks.

[illegible]

- *Staphylococcus aureus* (Gram +)
- *Staph. epidermidis*
- *Staph. saprophyticus*
- *Staph. aureus*

- PG: Lautstärke normalisieren
- PC: Stereo zu Mono (Zusammenfassen)
- PC: Stereo zusammenführen (Stereo-Mix)

OVERVIEW OF PCI(e) SUBSYSTEM

KISHON VIJAY ABRAHAM I, VIGNESH R

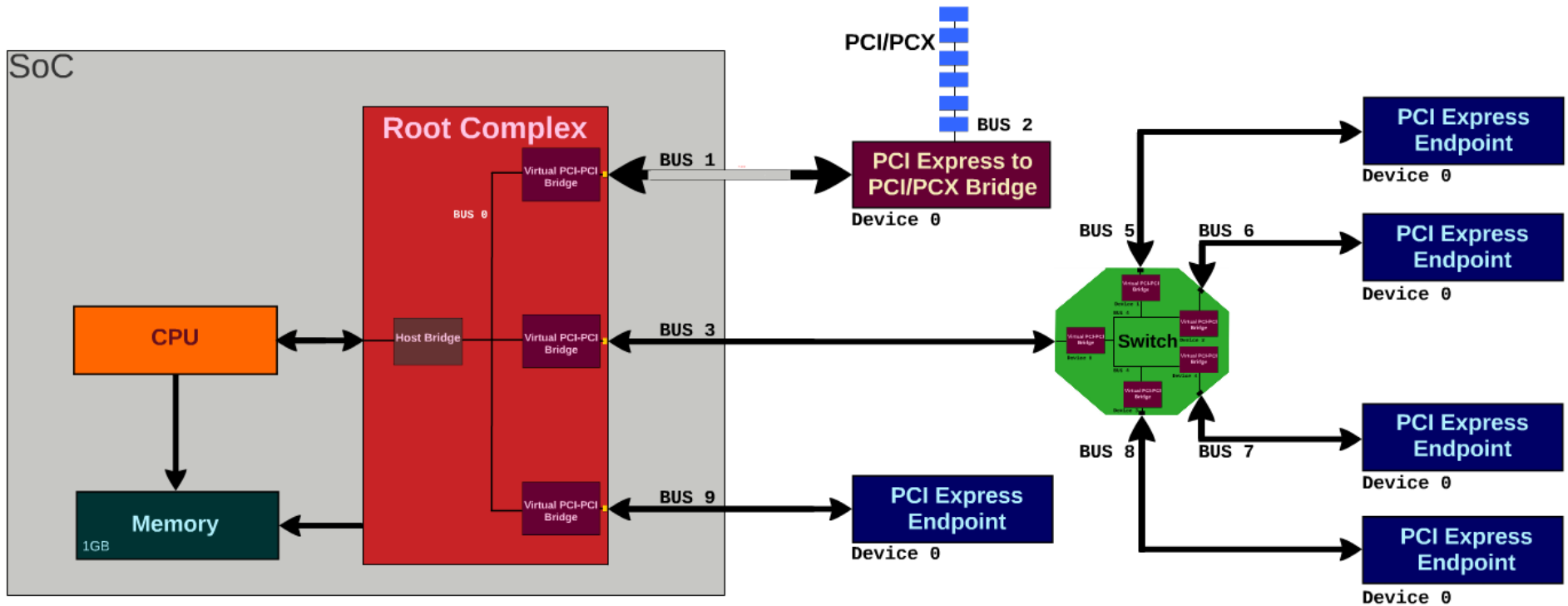
Introduction

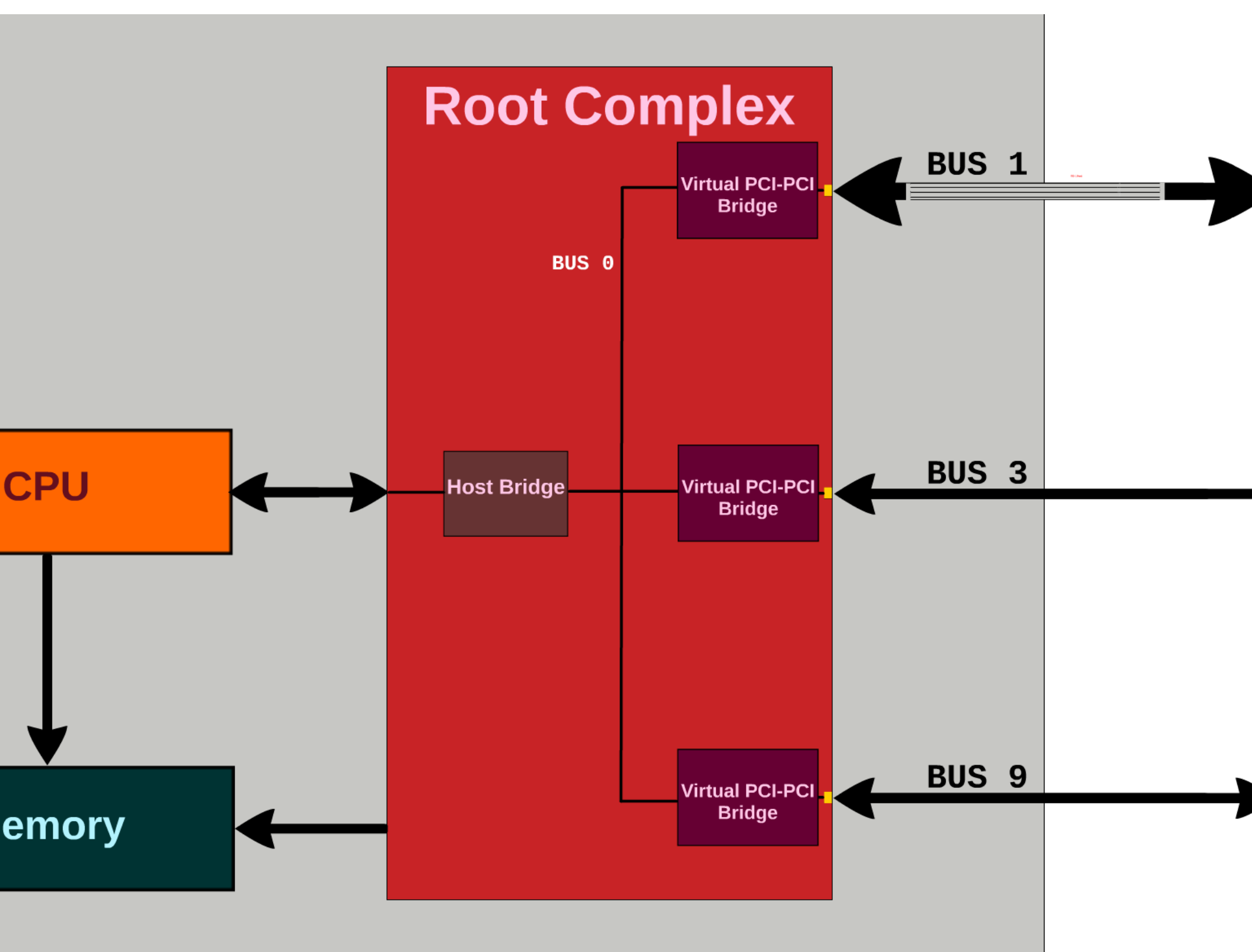
- PCIe Vs PCI
- Usage model and software interface same as PCI
- Serial communication interface like USB, SATA
- Performance: Higher throughput, multi-lane
- Add peripherals to the system (USB, Ethernet, SATA, DCAN etc..)

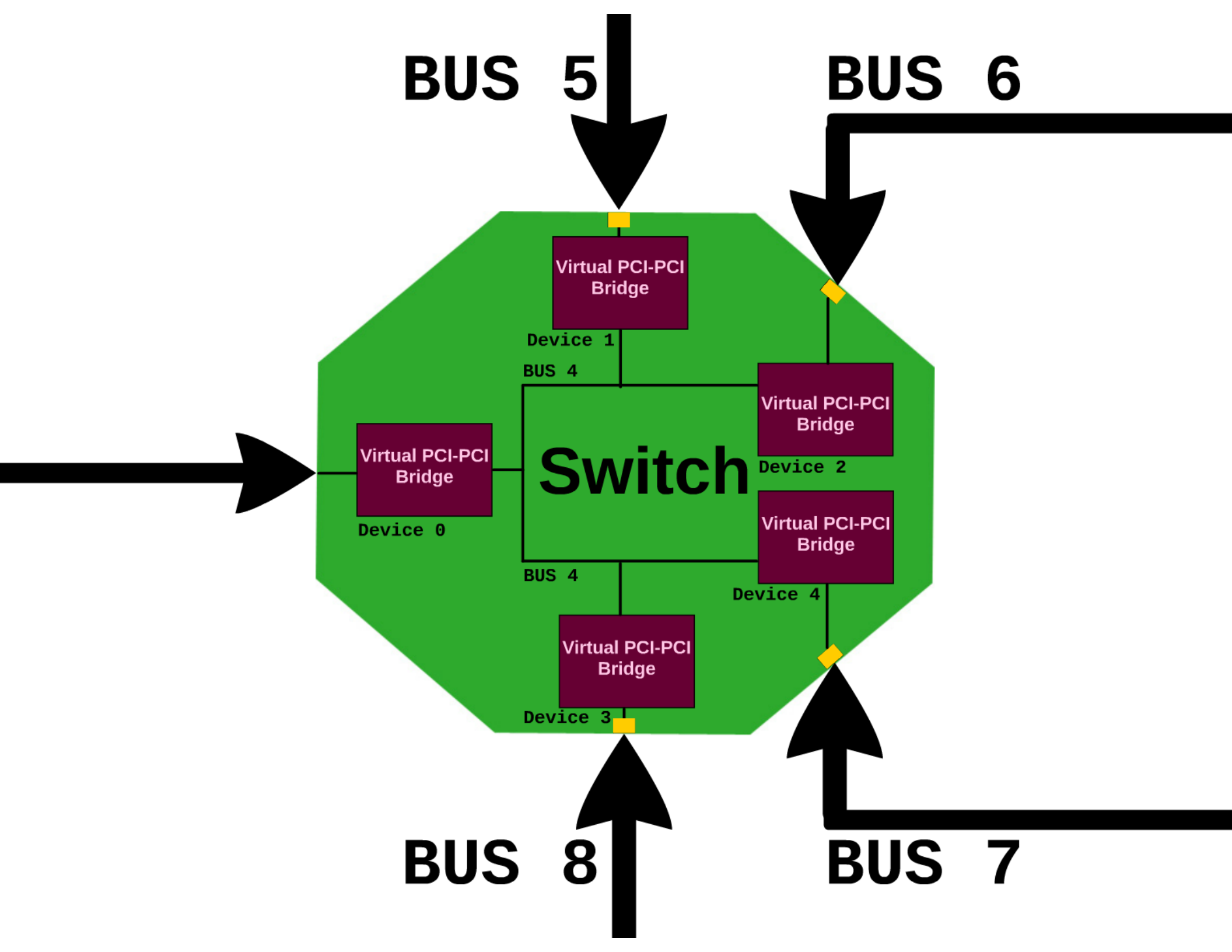
Terminology

- Root Complex - PCIe host
- Endpoint - PCIe device
- `cpu_addr` - CPU physical address (proc/iomem)
- `pci_addr` - PCI bus address
- Switches - allow more devices to be connected
- Enumeration - discovering devices

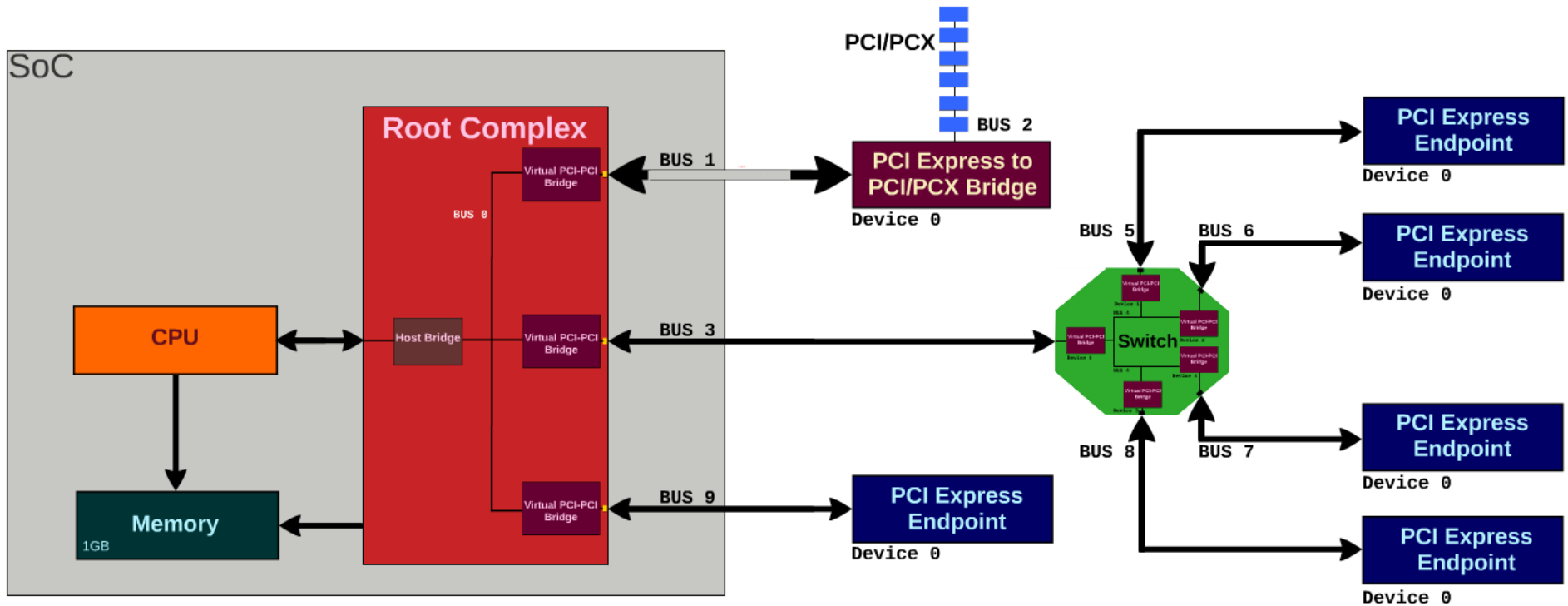
Topology







Topology



I/O Lines

RX+

RX-

TX+

TX-

REFCLK+

REFCLK-

PERST#

WAKE#

PRSNT1#

PRSNT2#

JTAG#

+12V#

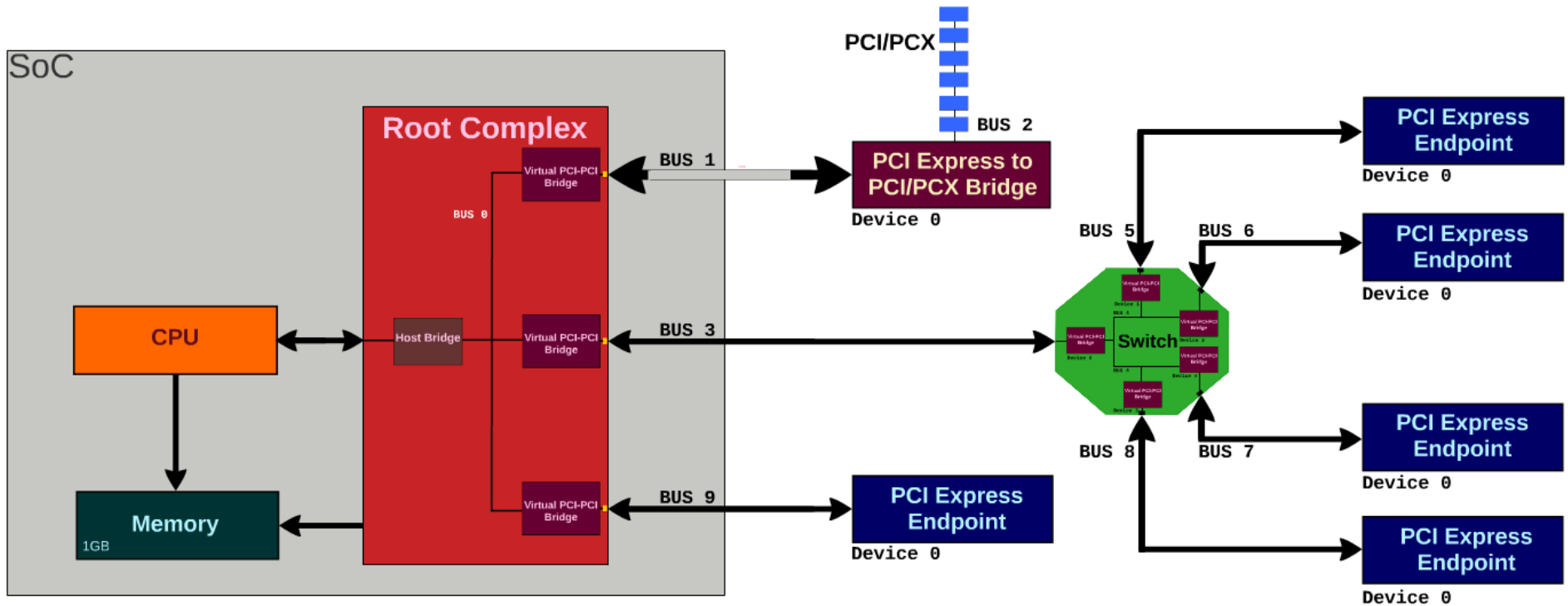
RX+

RX-

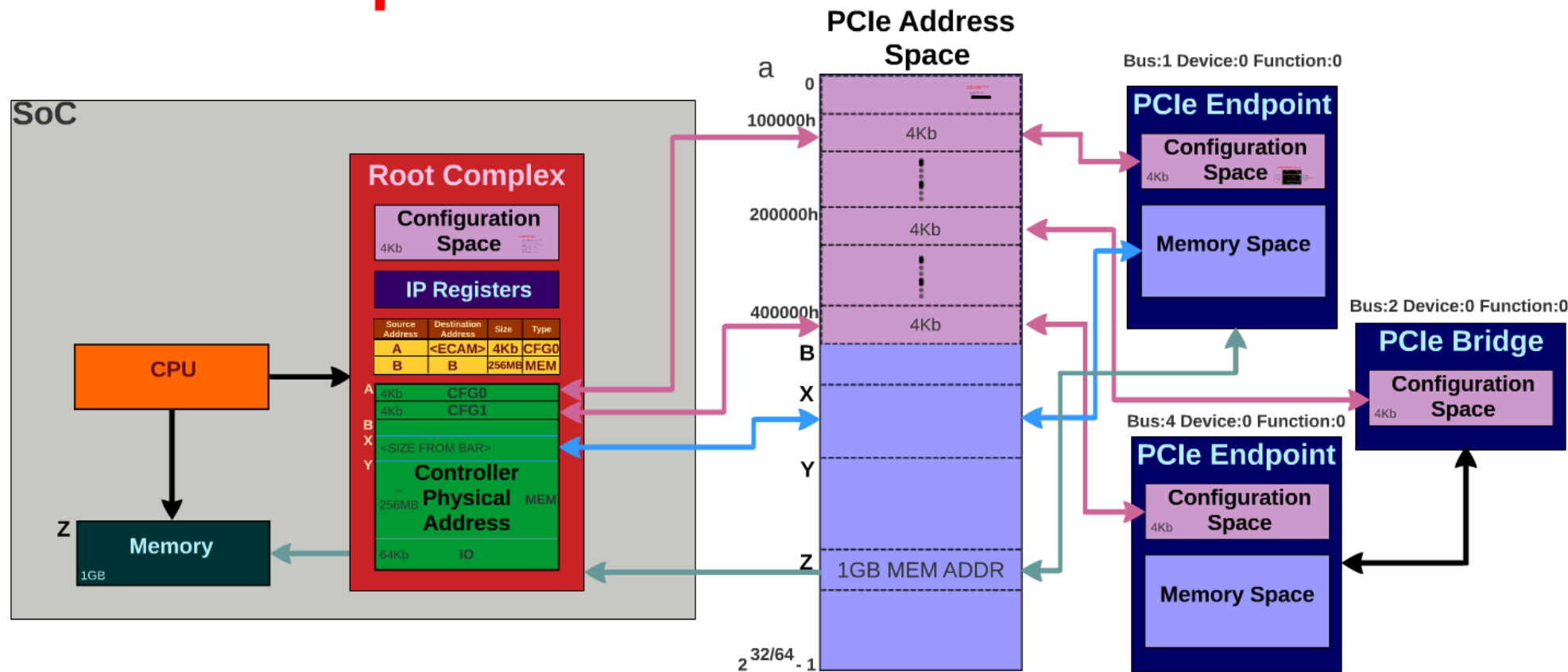
TX+

TX-

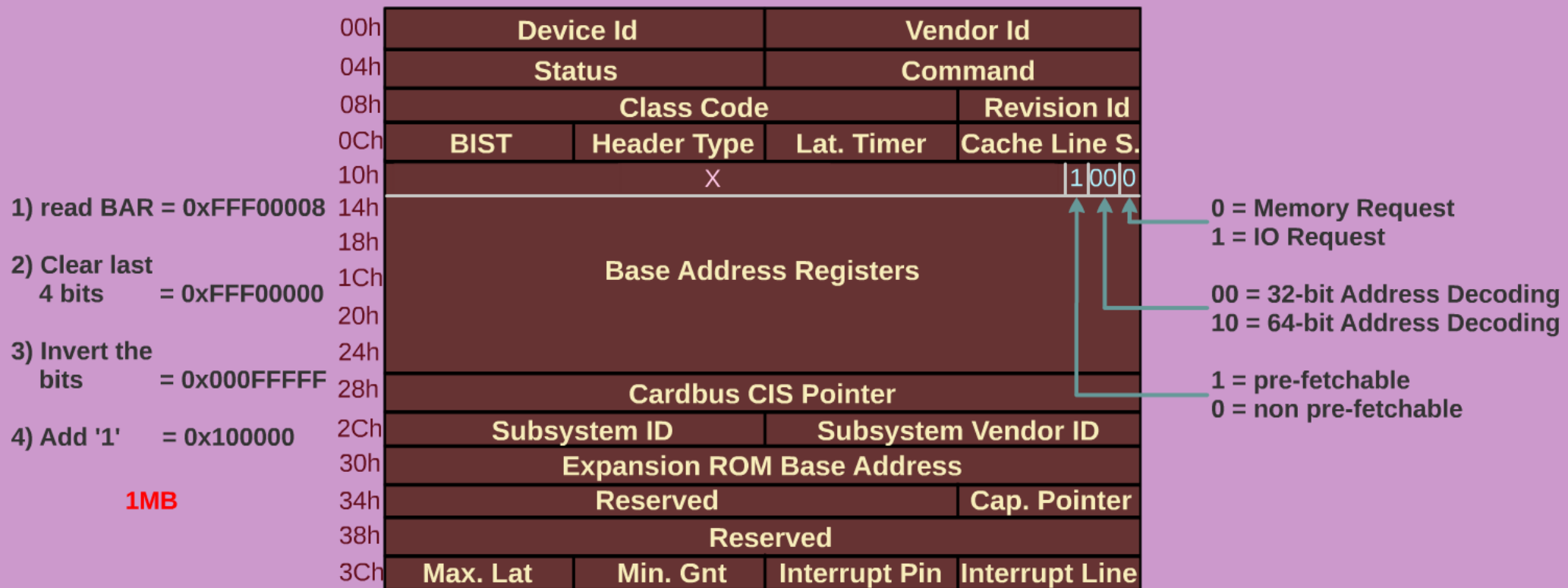
Topology



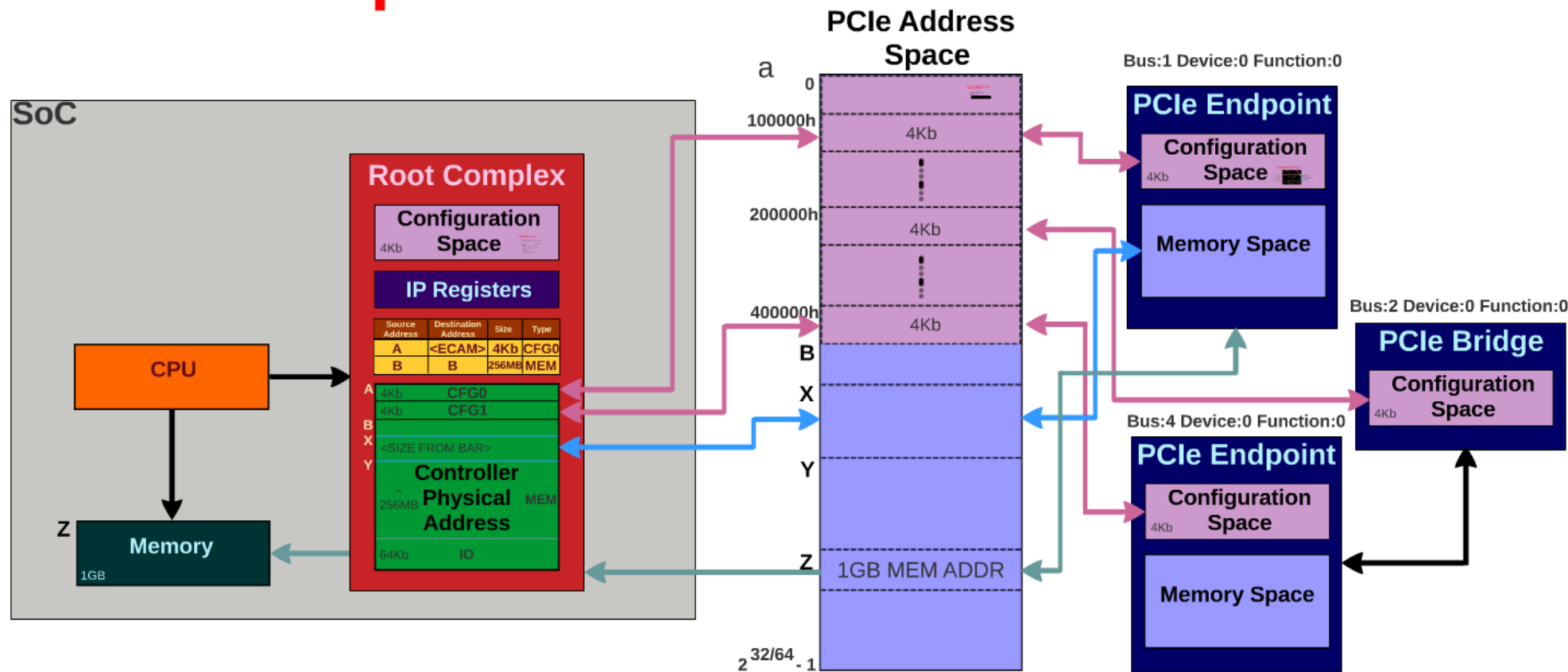
Address Space



Configuration Space Header

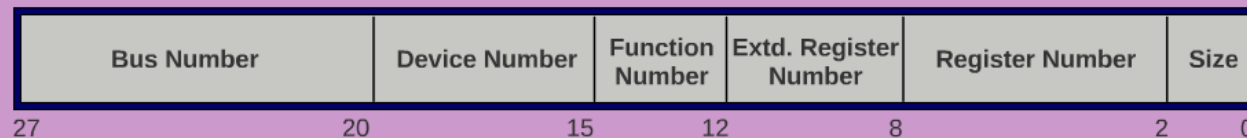


Address Space

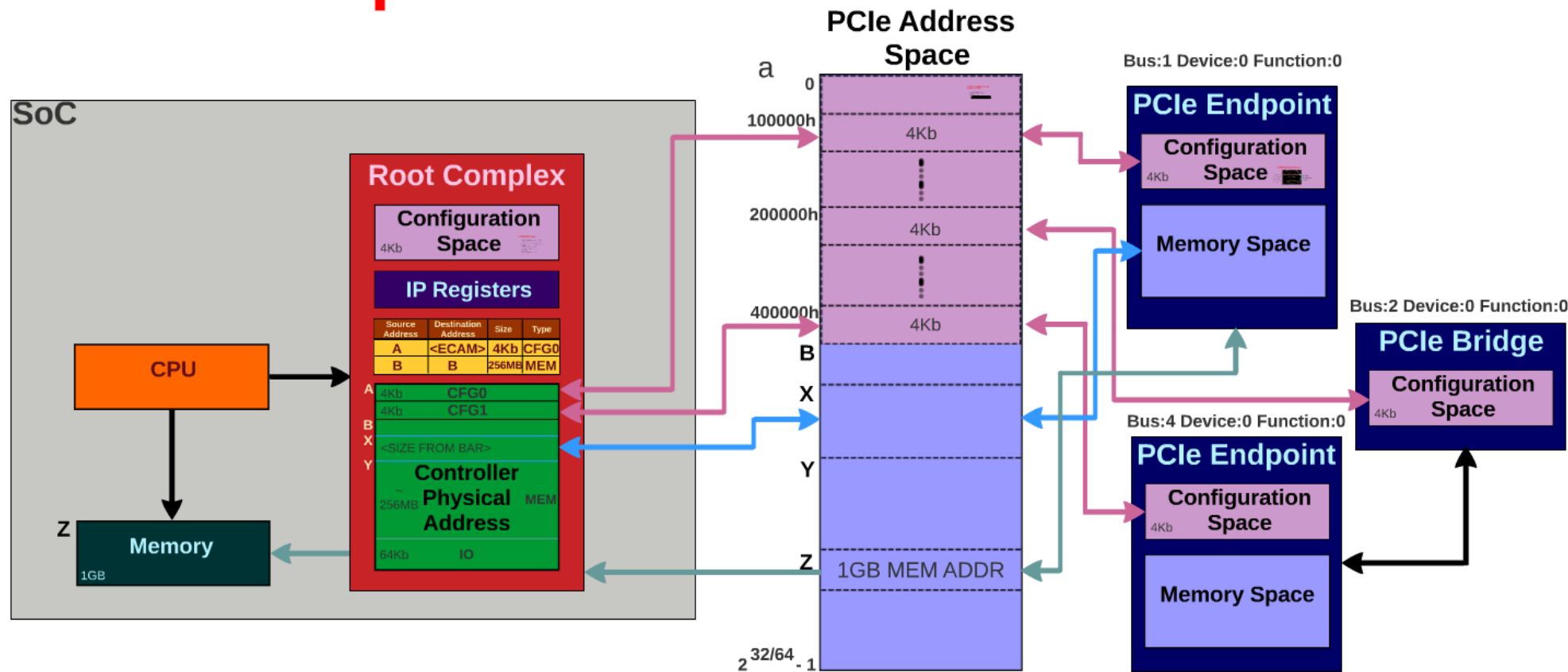


Enhanced Configuration Access Mechanism (ECAM)

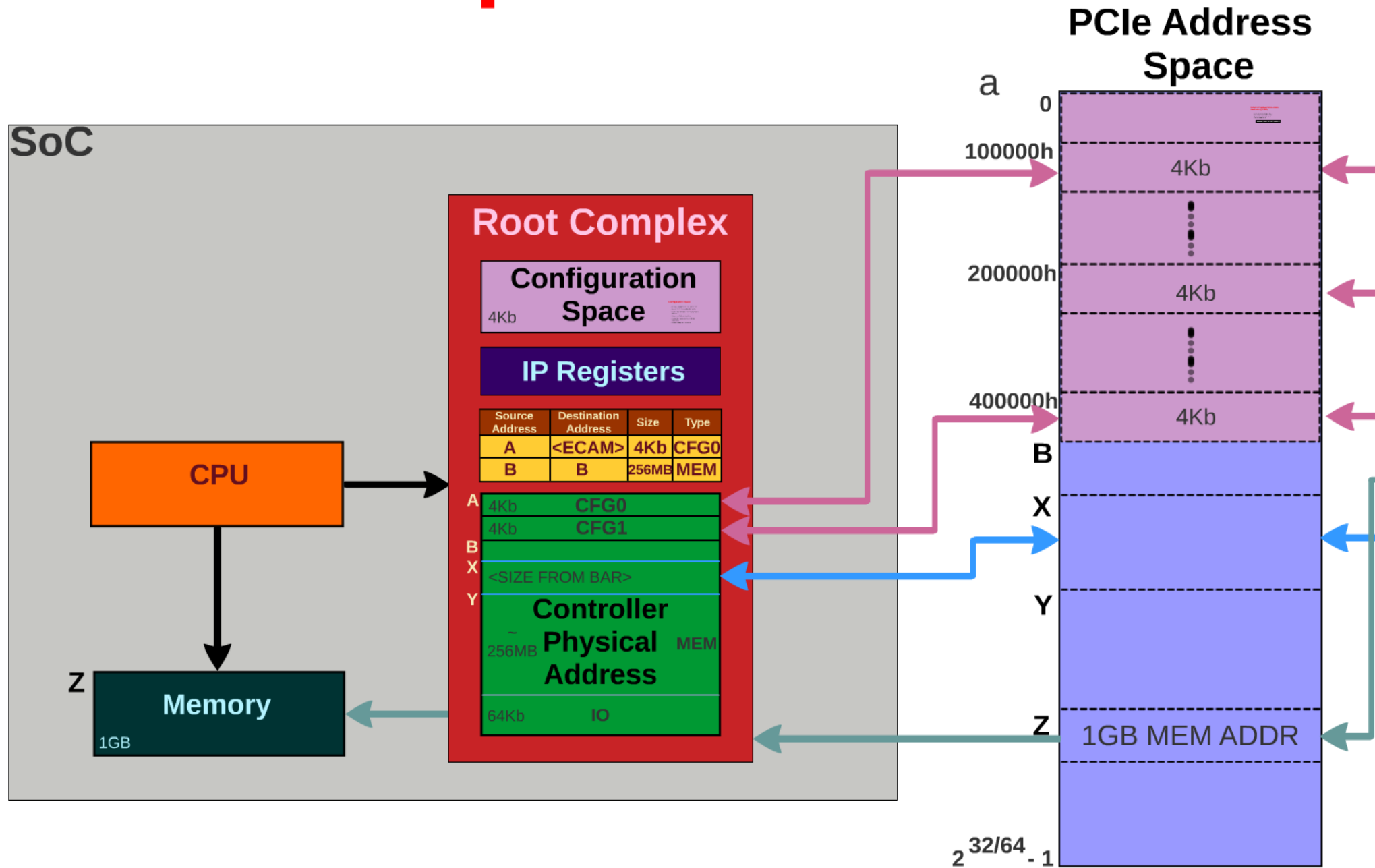
- Memory address (PCIe address space) determines configuration register accessed
- Function of bus number, device number, function and register number



Address Space



Address Space



PCIe Endpoint

Configuration Space

4Kb

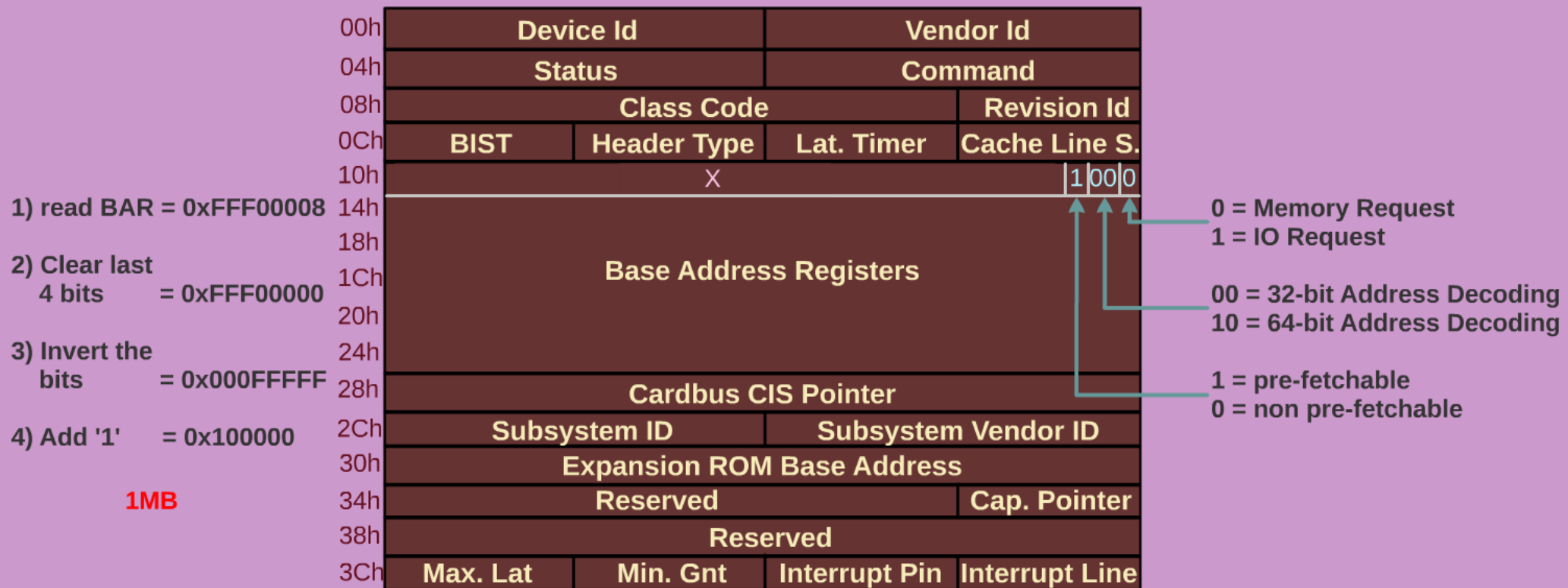
Configuration Space Header

Header 1	Header 2
Vendor ID	Device ID
Subsystem Vendor ID	Subsystem ID
Revision ID	Class Code
Base Address Register 1	Base Address Register 2
Base Address Register 3	Base Address Register 4
Base Address Register 5	Base Address Register 6
Base Address Register 7	Base Address Register 8
Base Address Register 9	Base Address Register 10
Base Address Register 11	Base Address Register 12
Base Address Register 13	Base Address Register 14
Base Address Register 15	Base Address Register 16

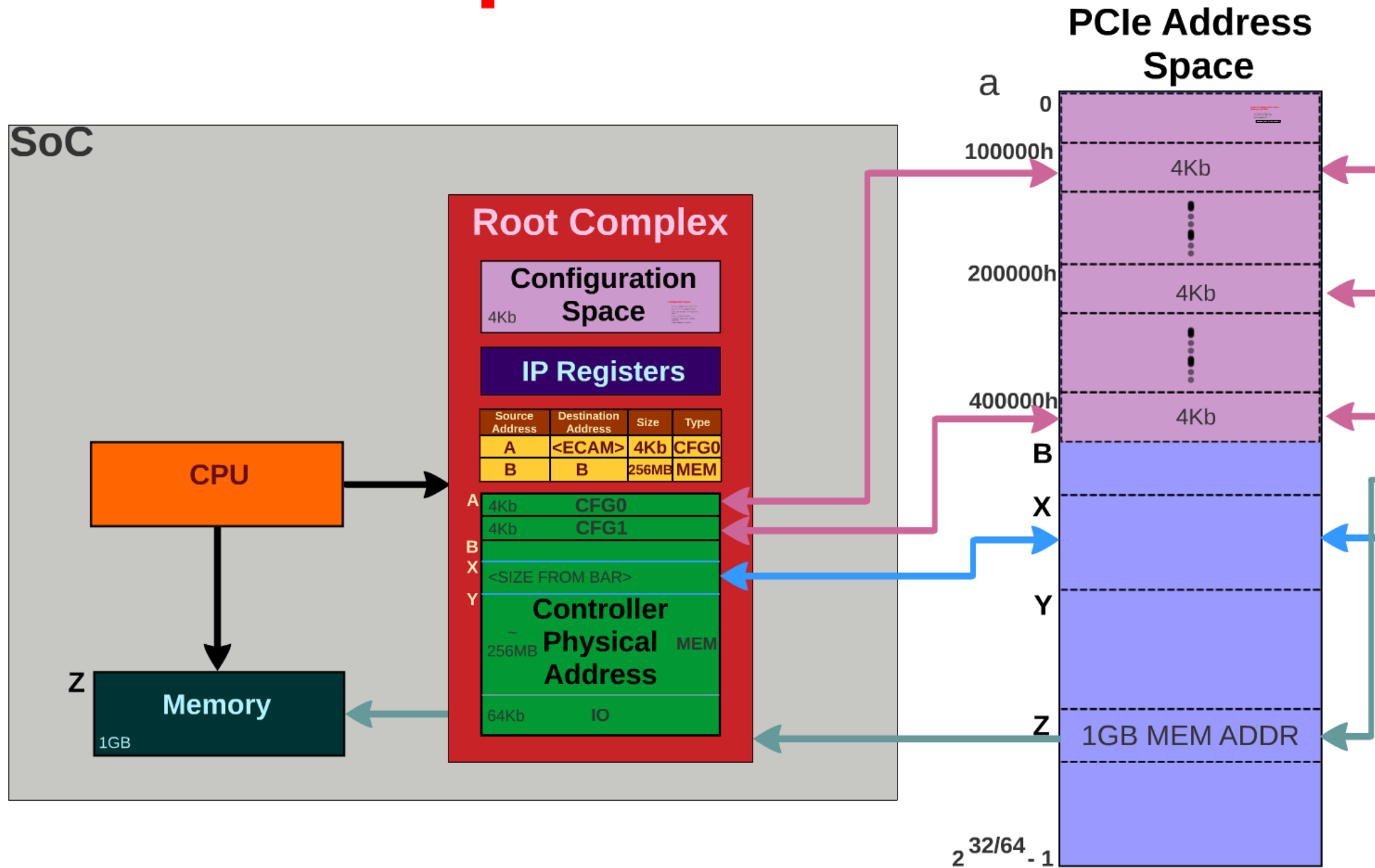
Memory Space

Bus:2

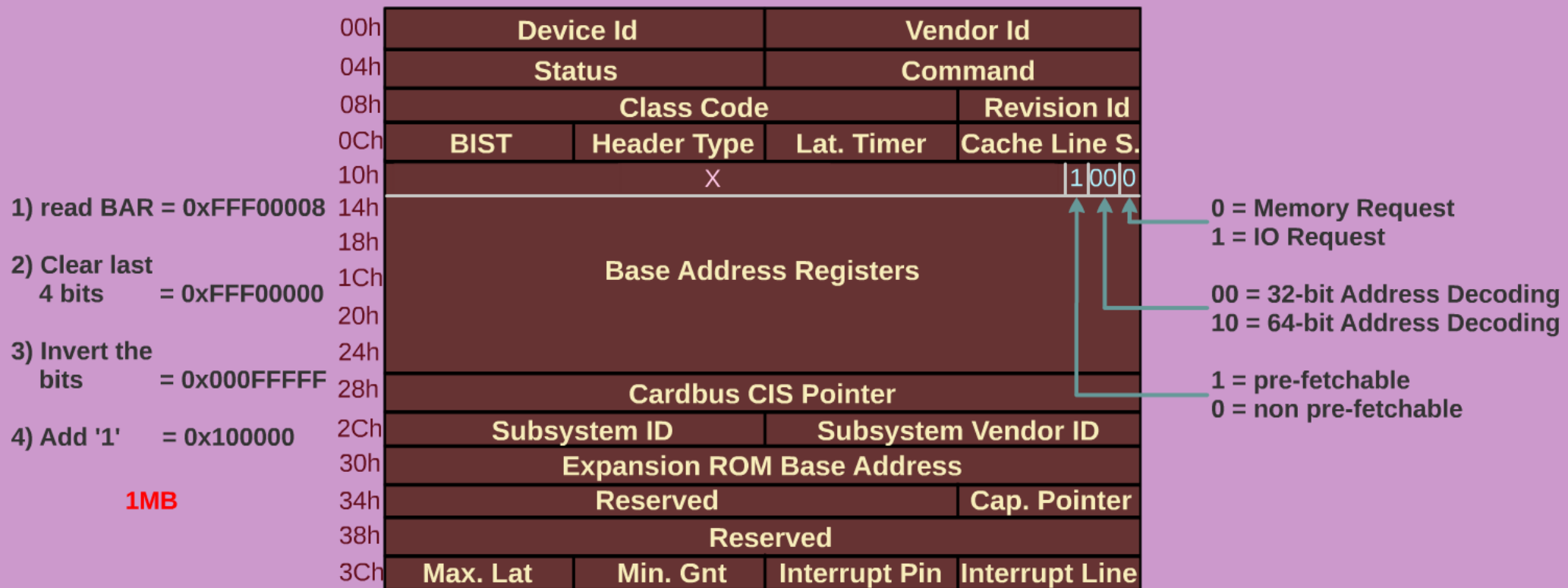
Configuration Space Header



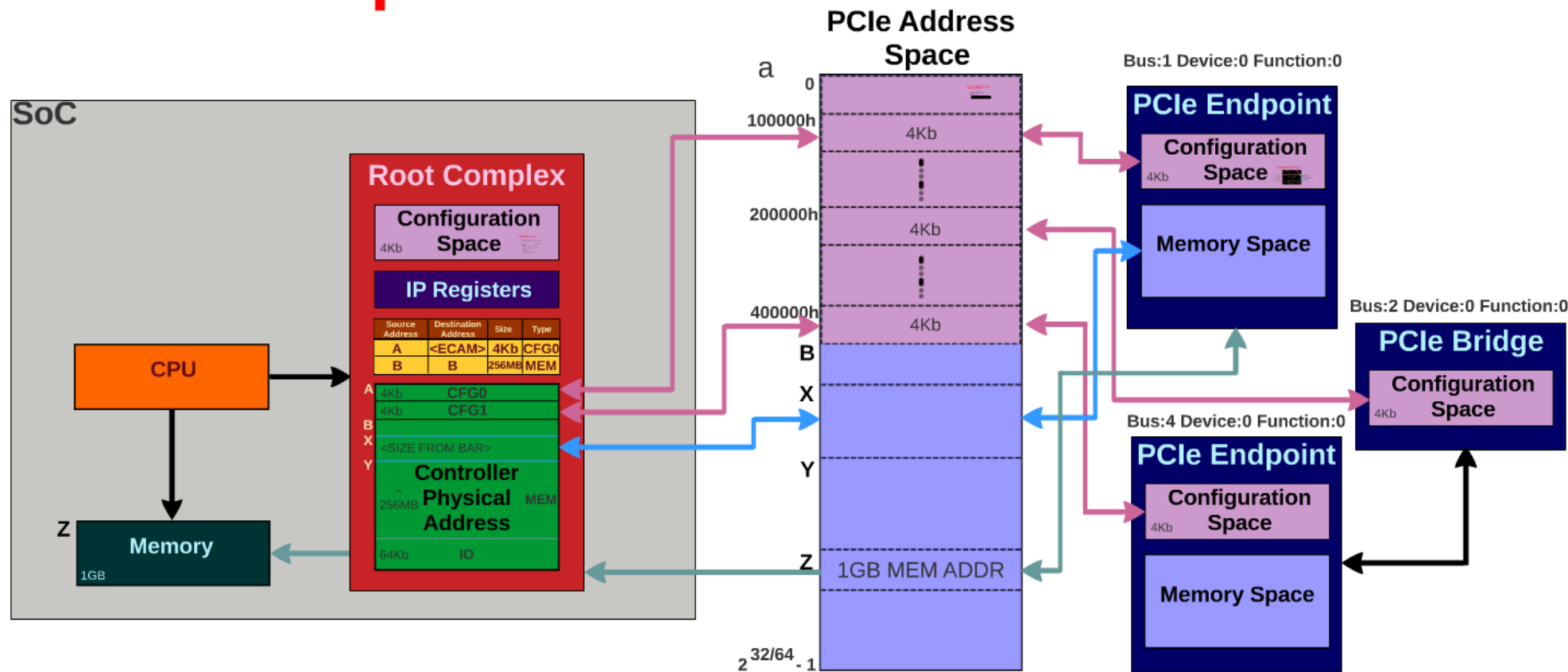
Address Space



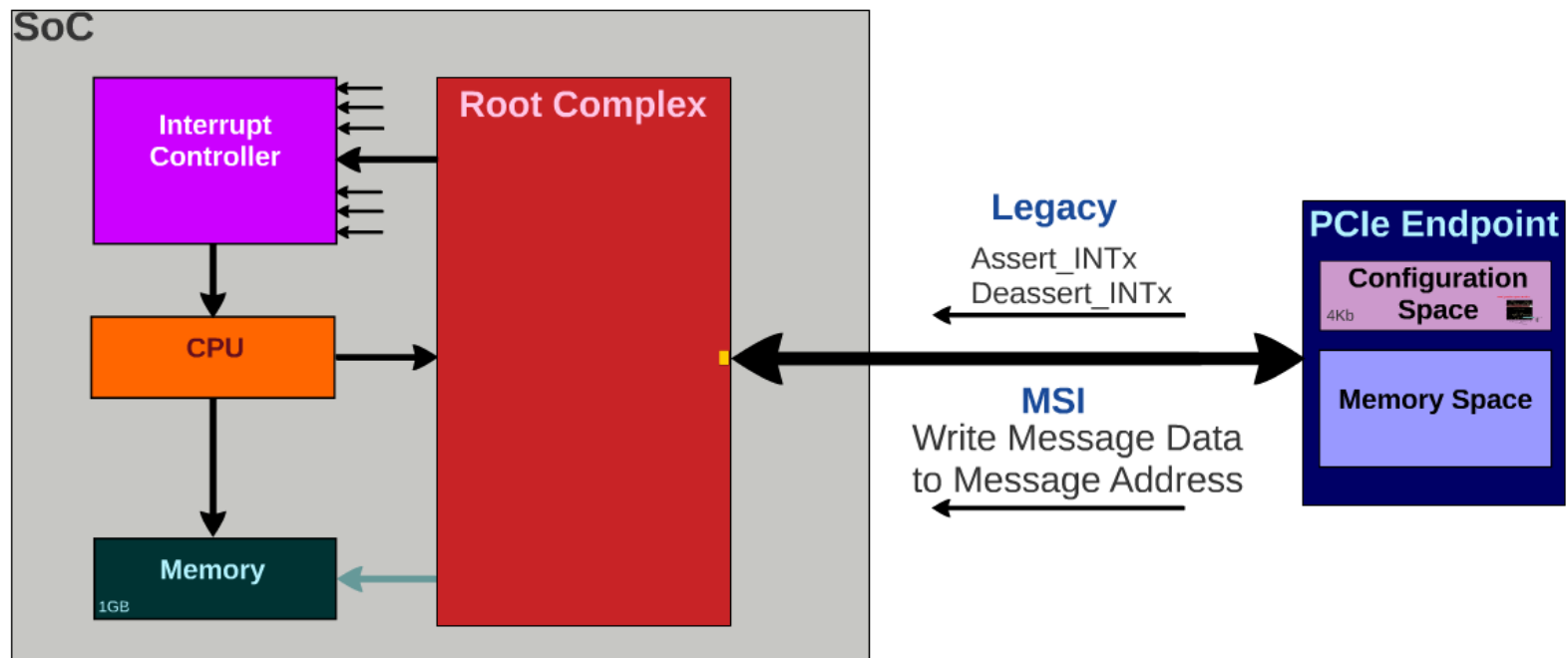
Configuration Space Header



Address Space



Interrupts



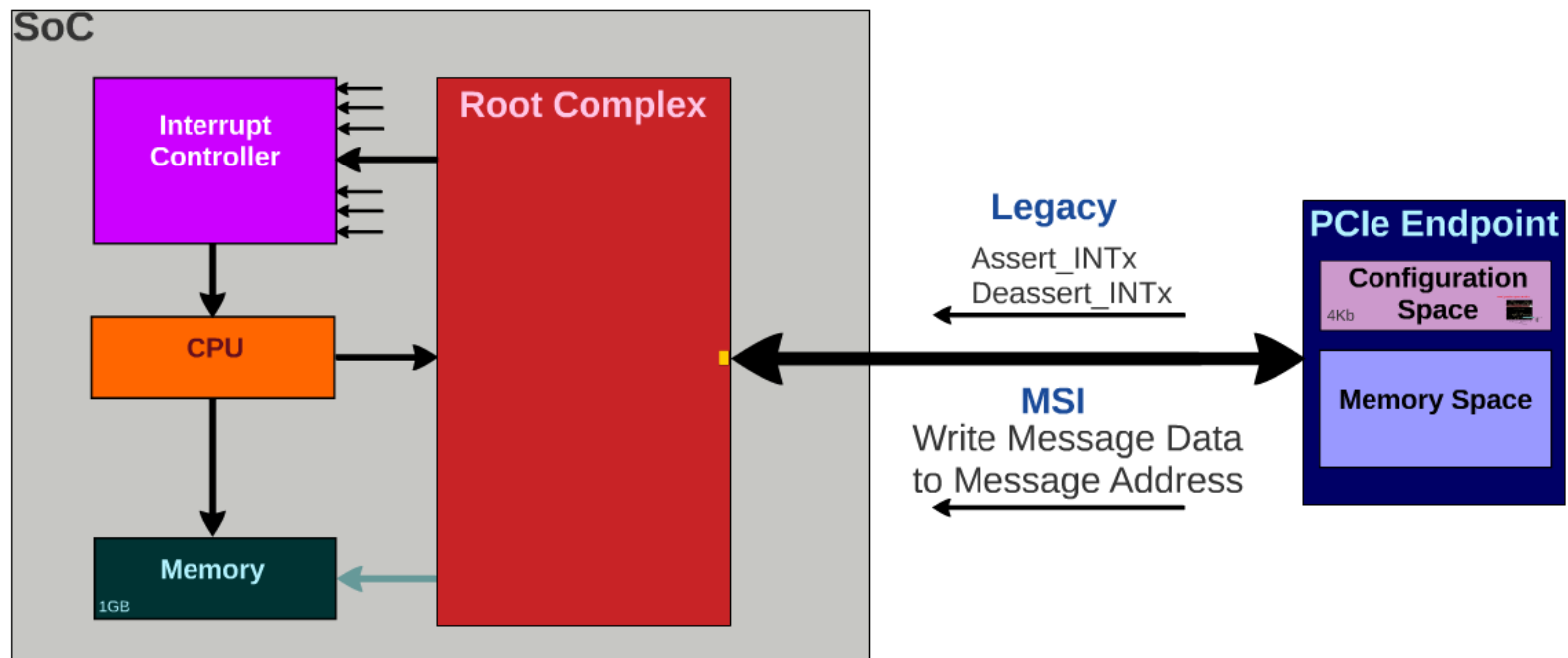
Configuration Space Header

00h	Device Id		Vendor Id		
04h	Status		Command		
08h	Class Code			Revision Id	
0Ch	BIST	Header Type	Lat. Timer	Cache Line S.	
10h	X				1 00 0
14h	Base Address Registers				
18h					
1Ch					
20h					
24h					
28h	Cardbus CIS Pointer				
2Ch	Subsystem ID		Subsystem Vendor ID		
30h	Expansion ROM Base Address				
34h	Reserved			Cap. Pointer	
38h	Reserved				
3Ch	Max. Lat	Min. Gnt	1	Interrupt Line	

0 = INTx Not used
 1 = INTA#
 2 = INTB#
 3 = INTC#
 4 = INTD#

Message Control	Next Pointer	Capability ID
Message Address		
Message Data		

Interrupts



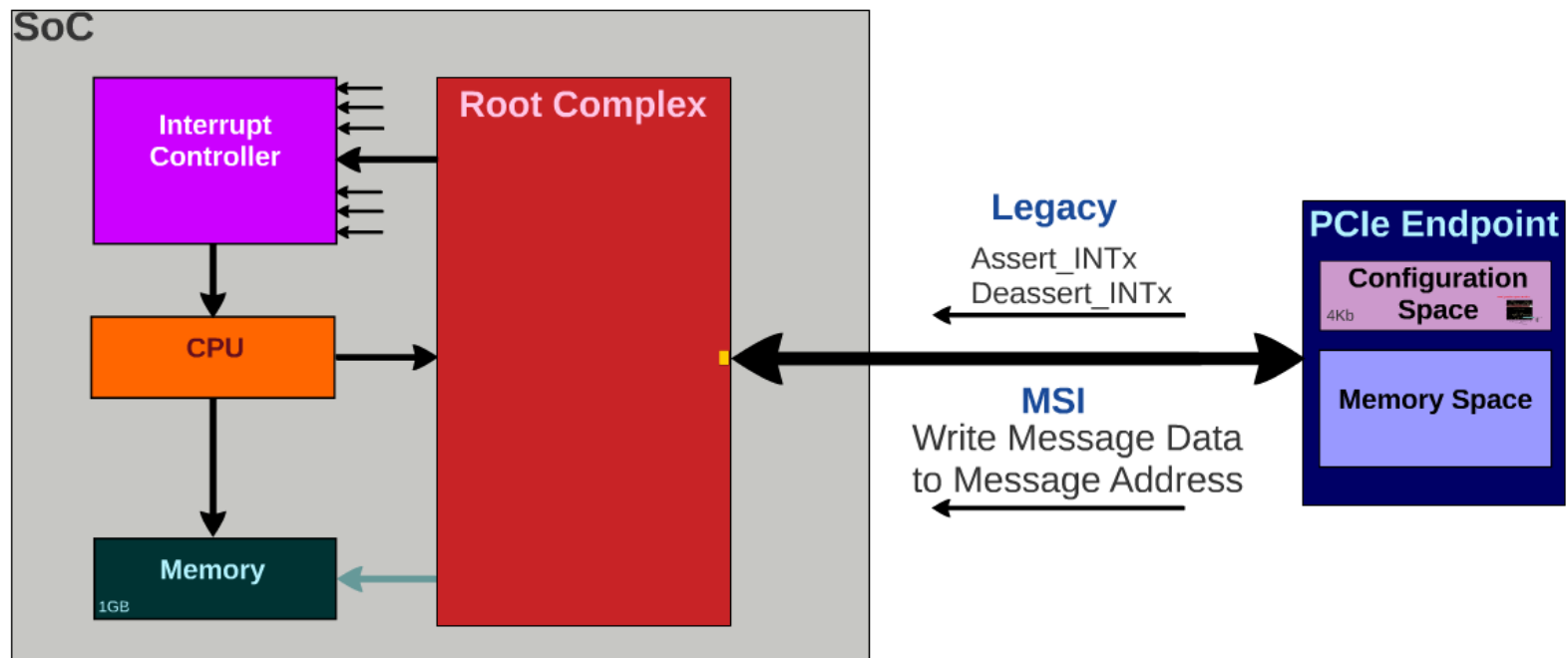
Configuration Space Header

00h	Device Id		Vendor Id		
04h	Status		Command		
08h	Class Code			Revision Id	
0Ch	BIST	Header Type	Lat. Timer	Cache Line S.	
10h	X				1 00 0
14h	Base Address Registers				
18h					
1Ch					
20h					
24h					
28h	Cardbus CIS Pointer				
2Ch	Subsystem ID		Subsystem Vendor ID		
30h	Expansion ROM Base Address				
34h	Reserved			Cap. Pointer	
38h	Reserved				
3Ch	Max. Lat	Min. Gnt	1	Interrupt Line	

0 = INTx Not used
 1 = INTA#
 2 = INTB#
 3 = INTC#
 4 = INTD#

Message Control	Next Pointer	Capability ID
Message Address		
Message Data		

Interrupts



Device Tree

```
pcie1: pcie@51000000 {
    compatible = "ti,dra7-pcie";
    reg = <0x51000000 0x2000>, <0x51002000 0x14c>, <A 0x2000>;
    reg-names = "rc_dbics", "ti_conf", "config";
    interrupts = <0 232 0x4>, <0 233 0x4>;
    device_type = "pci";
    #address-cells = <3>;
    #size-cells = <2>;
    ranges = <0x82000000 0 B B 0 0xfffd000>;
    interrupt-map-mask = <0 0 0 7>;
    interrupt-map = <0 0 0 1 &pcie1_intc 1>,
                    <0 0 0 2 &pcie1_intc 2>,
                    <0 0 0 3 &pcie1_intc 3>,
                    <0 0 0 4 &pcie1_intc 4>;
    pcie1_intc: interrupt-controller {
        interrupt-controller;
        #address-cells = <0>;
        #interrupt-cells = <1>;
    };
};
```

'ranges' Property

```
#address-cells = <3>;
#size-cells = <2>;
ranges = <0x82000000 0 B B 0 0xfffd000>;
```

PCI Address

Size

CPU Address

n p t 0 0 0 s s

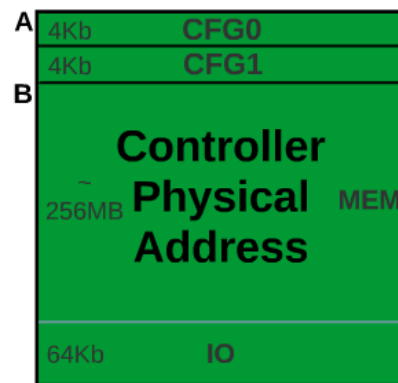
00: Configuration Space
01: I/O Space
10: 32 bit Memory Space
11: 64 bit Memory Space

prefetchable (cacheable)
relocatable

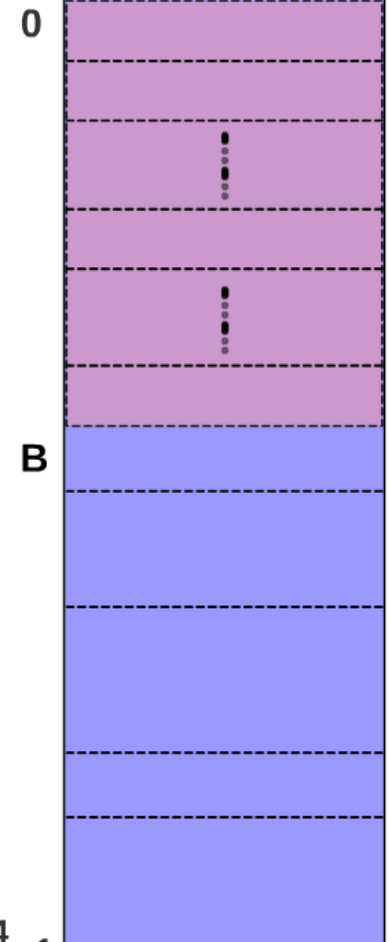
CPU

PCIE

Source Address	Destination Address	Size	Type
B	B	~256MB	MEM



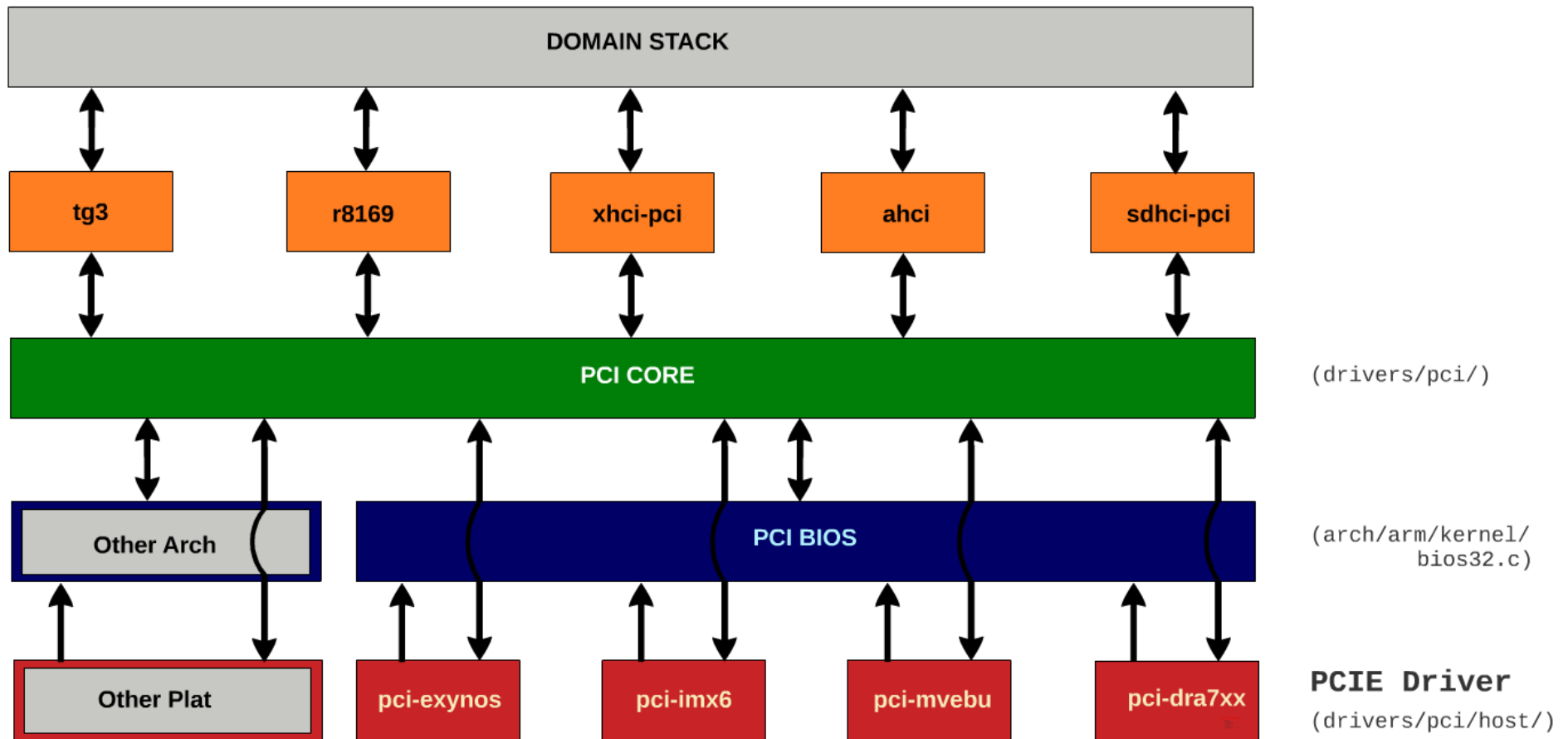
PCIe Address Space



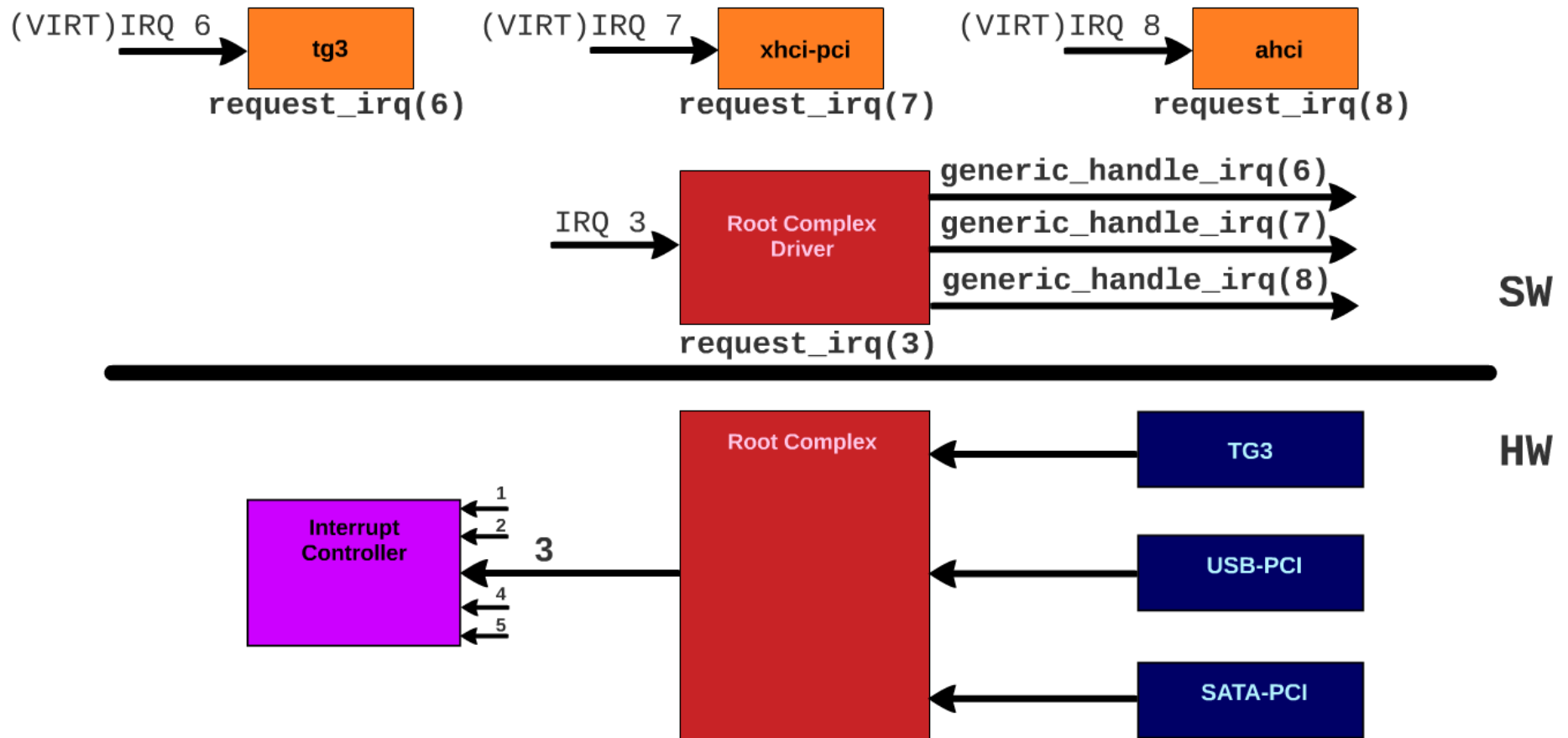
Device Tree

```
pcie1: pcie@51000000 {
    compatible = "ti,dra7-pcie";
    reg = <0x51000000 0x2000>, <0x51002000 0x14c>, <A 0x2000>;
    reg-names = "rc_dbics", "ti_conf", "config";
    interrupts = <0 232 0x4>, <0 233 0x4>;
    device_type = "pci";
    #address-cells = <3>;
    #size-cells = <2>;
    ranges = <0x82000000 0 B B 0 0xfffd000>;
    interrupt-map-mask = <0 0 0 7>;
    interrupt-map = <0 0 0 1 &pcie1_intc 1>,
                    <0 0 0 2 &pcie1_intc 2>,
                    <0 0 0 3 &pcie1_intc 3>,
                    <0 0 0 4 &pcie1_intc 4>;
    pcie1_intc: interrupt-controller {
        interrupt-controller;
        #address-cells = <0>;
        #interrupt-cells = <1>;
    };
};
```

Linux PCI(e) Subsystem



Interrupt Handling



lspci

- Displays all PCI buses, devices in a system

```
root@am57xx-evm:~# lspci
00:00.0 PCI bridge: Texas Instruments Device 8888 (rev 01)
01:00.0 PCI bridge: Pericom Semiconductor Device 2304 (rev 05)
02:01.0 PCI bridge: Pericom Semiconductor Device 2304 (rev 05)
02:02.0 PCI bridge: Pericom Semiconductor Device 2304 (rev 05)
03:00.0 SATA controller: ASMedia Technology Inc. ASM1062 Serial
ATA Controller (rev 01)
04:00.0 USB controller: Etron Technology, Inc. EJ168 USB 3.0 Host
Controller (rev 01)
```

- http://linuxcommand.org/man_pages/lspci8.html

Acknowledgements

- Jingoo Han, Pratyush Anand, Bjorn Helgaas
- Linux Community
- Texas Instruments
- Linux Foundation

References

- PCI Local Bus Specification 3.0
- PCI Express Base Specification 3.0
- PCI Express System Architecture (by Mindshare)

Happy Hacking!

Feedback:
kishon@ti.com
kishonvijayabraham@gmail.com

vigneshr@ti.com
vignesh.r.blr@gmail.com